Parallelized simulation code for multiconjugate adaptive optics

Aron J. Ahmadia, Brent L. Ellerbroek

Aron J. Ahmadia, Brent L. Ellerbroek, "Parallelized simulation code for multiconjugate adaptive optics," Proc. SPIE 5169, Astronomical Adaptive Optics Systems and Applications, (24 December 2003); doi: 10.1117/12.506553

SPIE.

Event: Optical Science and Technology, SPIE's 48th Annual Meeting, 2003, San Diego, California, United States

Parallelized Simulation Code for Multiconjugate Adaptive Optics

A.J. Ahmadia[a] and B.L. Ellerbroek[b]

[a]Gemini Observatory, 670 N. A'ohoku Place, Hilo, HI 96720
[b]AURA New Initiatives Offices, 950 N. Cherry Avenue, Tucson, AZ 85719

ABSTRACT

Advances in adaptive optics (AO) systems are necessary to achieve optical performance that is suitable for future extremely large telescopes (ELTs). Accurate simulation of system performance during the design process is essential. We detail the current implementation and near-term development plans for a coarse-grain parallel code for simulations of multiconjugate adaptive optics (MCAO). Included is a summary of the simulation's computationally intensive mathematical subroutines and the associated scaling laws that quantify the size of the computational burden as a function of the simulation parameters. The current state of three different approaches to parallelizing the original serial code is outlined, and the timing results of all three approaches are demonstrated. The first approach, coarse-grained parallelization of the atmospheric propagations, divides the tasks of propagating wavefronts through the atmosphere among a group of processors. The second method of parallelization, fine-grained parallelization of the individual wavefront propagations, is then introduced. Finally, a technique for computing the wavefront reconstructions is analyzed. A parallel version of the block-symmetric Gauss-Seidel smoother, used in the conjugate-gradients reconstructor with multigrid-solver preconditioning, has been implemented. The timing results demonstrate that this is currently the fastest known full-featured, operational multiconjugate adaptive optics simulation.

Keywords: Adaptive optics, propagation simulations, parallel computing, Beowulf cluster

1. INTRODUCTION

Future advances in the theory and development of adaptive optics (AO) systems will be fundamental for achieving the desired levels of image quality and optical performance from future giant astronomical telescopes with aperture diameters of 30 meters or more[1,2,3]. An important aspect of developing these new adaptive optics technologies and systems is the ability of the scientists and designers to accurately simulate system performance during the design process to establish an optimized set of first-order design parameters, and develop error budgets for the impact of implementation error sources and higher-order effects. The adaptive optics simulation code, previously introduced in 2001[4], is a well-developed model for studying current and future adaptive optics systems. Such systems include multiconjugate adaptive optics (MCAO), which employ multiple deformable mirrors (DM) and wavefront sensors (WFS) to compensate for the effects of atmospheric turbulence across extended fields of view[5,6,7,8,9].

In an effort to reduce execution times for complex adaptive optics systems, particularly for very high order MCAO systems on future giant telescopes, we have introduced parallel code into the simulation. The sections of the code that were the most CPU-intensive have been identified and parallelized. A combination of C and compiled Matlab code has been used to implement a parallel solution using several mathematical and parallel packages. The code has been tested against the original serial version for speed and verification of data. The serial code's simple yet flexible interface allowed a novice Matlab user to wield the full power of the simulation. During the parallelization process, much of this simplicity and flexibility was maintained. The parallel design was crafted such that improvements in speed will scale well from a small x86 cluster to large supercomputers.

The simulation code has four phases: (1) an interactive command menu that allows the programmer to input the parameters of the simulation; (2) preprocessing to compute the full description of the adaptive optics system from the specified parameters; (3) the actual simulation loop that evaluates the effects of the atmosphere, telescope, and adaptive optics system on a set of wavefronts over a range of sequential time steps; and (4) postprocessing analysis of the results. The parameters of the simulation allow for multiple atmospheric layers, geometrical or diffraction models for optical propagation, the introduction of various types of measurement and correction error, and the ability to simulate laser guide stars and multiconjugate adaptive optics systems.

Astronomical Adaptive Optics Systems and Applications, edited by Robert K. Tyson, Michael Lloyd-Hart, Proceedings of SPIE Vol. 5169 (SPIE, Bellingham, WA, 2003)
0277-786X/03/$15 · doi: 10.1117/12.506553

Details about the methods and results are presented in the subsequent sections. Section 2 elaborates on the simulation and its capabilities. Section 3 identifies the pieces of the simulation that are the most time-intensive, and introduces the theoretical scaling for simulations of extremely large telescopes. Section 4 describes the basic programming techniques used to add parallel code to the software. Sections 5 and 6 fully detail the decisions made and the improvements in execution time obtained from parallelizing the code for the two major elements of the simulation–the wavefront propagator and the reconstructor. Section 7 summarizes the findings.

1. SIMULATION DESCRIPTION

This section provides an overview of the serial simulation that has been previously introduced and described[4]. Figure 1 illustrates the basic structure of the software and steps of the simulation.

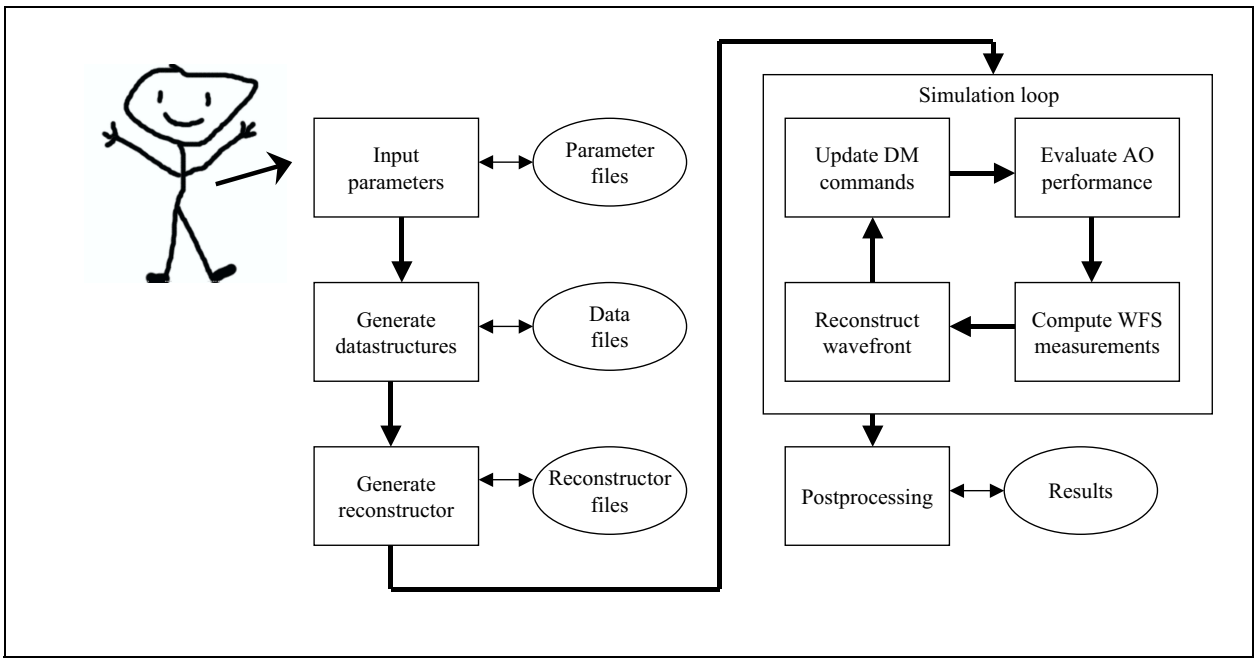


Figure 1: Serial Simulation Overview

The software's interactive menu allows the user to generate a custom scenario for simulation. The menu of choices available to define the system include the atmospheric model, modes of performance evaluation, defining optical surfaces, wavefront sensor and guide star parameters, deformable and tip-tilt mirror parameters, and the control algorithm parameters. The user is also able to customize additional parameters that allow for multiple atmospheric layers, flexible grid sizes, geometrical or diffraction models for optical propagation, the introduction of various types of measurement and correction error, and the ability to simulate laser guide stars and MCAO systems.

Once the system has been defined, the simulation generates necessary data structures. These include mirror figure errors, atmospheric phase screens, aperture masks, wavefront sensor subaperture locations, deformable mirror actuator grid locations, and the corresponding mirror-to-sensor geometric influence matrix. Based upon the turbulence profile and this influence matrix, a wavefront reconstruction matrix is computed to minimize field and aperture averaged mean-square residual wavefront error over the user's chosen directions for system performance evaluation.

Once the user has configured the simulation problem, the program enters a loop-driven model of the adaptive optics system's behavior in the time domain. Sequentially, the simulation propagates wavefronts from guide stars and evaluation points down through the atmosphere and the telescope's corrective mirrors using the chosen model for

optical propagation. The wavefront sensor measurements for the guide stars are calculated and read out. The actuator commands to the deformable and tip-tilt mirrors are determined using a vector-matrix-multiply wavefront reconstructor, which in the near future will be upgraded to a more computationally efficient method[10,11] to enable simulations of MCAO systems on giant telescopes. The reconstructed actuator errors are then filtered by the finite difference equation that determines the temporal dynamics of the adaptive optics control loop. The performance of the adaptive optics system is then measured by evaluating the mean-square phase variance, point spread function, and Strehl ratio for each of the wavefronts from the evaluation fields after propagation.

The loop is repeated over a finite set of time quantum, which is usually an interval that is sufficiently long for the adaptive optics system to stabilize. After the simulation has concluded, the program performs some postprocessing. Postprocessing includes calculating the mean point spread functions and Strehl ratio for each evaluation direction and wavelength.

1. COMPUTATIONAL SCALING

As noted earlier[4], a reasonably fast simulation of MCAO for ELTs should be feasible, in principle, if the computationally intensive sections of the simulation are implemented in parallel. To identify the areas of the simulation that dominated the CPU time, the Matlab Profiler was run on a typical simulation using simple geometric ray-tracing propagation for a 24-meter case. The results showed that nearly 50% of the simulation time was spent in the propagation of wavefronts through the atmosphere and optical system of the telescope. Another 45% of the simulation time spent in the wavefront reconstructor. These two sections of the code were selected for mathematical scaling analysis.

The standard size of propagation grid that is currently used for simulations of 8-meter class telescopes is 50 meters. Assuming a turbulence outer scale of about 30 meters, the grid size for 30-meter class telescopes does not need to be much greater than 60 meters[4]. Although the grid sizes for optical propagations do play an important role in simulations of extremely large telescopes, the increase in grid size of 20% will yield marginal scaling effects. The computation requirements for these propagations scale as $n^2 \log n$, where n is the number of points across the propagation grid.

The serial wavefront reconstructor's most efficient implementation has been previously detailed[10,11]. This approach takes advantage of the sparsity of the deformable mirror-to-wavefront sensor influence matrix to solve the system using iterative techniques. The algorithm applies the conjugate-gradient method with multigrid block-symmetric Gauss-Seidel smoothing iterations for preconditioning. The order of operations scales empirically as about $O(n^{5/4})$, with n representing the number of wavefront sensor measurements. For a 32-meter telescope with 4 times the diameter of an 8-meter telescope, it is estimated that 16 times as many actuators and subapertures would be required. This corresponds to an n 16 times greater, and an overall effect of about 32 on the amount of computations necessary. For comparison, the computational requirements for the standard vector-matrix-multiply reconstructor would increase by about a factor of $16^2 = 256$, assuming the reconstruction matrix could even be computed for a system of this size.

It is apparent that wavefront propagation and the wavefront reconstructor are the primary drivers for CPU time for extremely large telescopes in the 30-meter class and above. The rest of this paper details our efforts to generate parallel implementations of the existing Matlab simulation code to reduce the overall simulation time.

1. IMPLEMENTING PARALLEL MATLAB CODE USING MPI AND C

The serial simulation was originally written in Matlab for flexibility and ease of use. We seek to develop a parallel version of the code that retains these features, to the greatest extent possible, while minimizing hardware and software costs and the scope of the software development effort. Our approach employs a Beowulf cluster based on the MPI communications protocol. High-level blocks of the code with minimum interprocess communication (separate wavefront propagations) can be effectively parallelized as independent compiled Matlab tasks. Tasks that require higher levels of interprocess communications, such as the wavefront reconstructions and individual propagations, are parallelized using optimized parallel mathematical packages.

4.1 Background Parallel Theory

The integral premise of parallelization is to deliver improved overall speed of a computing task by increasing the number of processors utilized rather than improving the speed of an individual processor.

The hardware architecture of a parallel computer is based on either a shared memory or distributed memory paradigm. In the shared memory architecture, individual processor units are mounted on the same memory bus cooperatively accessing and writing to the same memory regions equally. Within a distributed memory architecture, individual processors and memory banks are interconnected usually via high-speed network connections. One such implementation is the Beowulf cluster, a group of workstation computers connected over a fast network connection, which uses message passing to perform computational work in parallel.

Serial code migrates to parallel architectures in different ways. The coarse-grain technique involves splitting up the work at high levels of the code, at its most outer loops. Large individual tasks are completed on each processing node, with little or no data interaction during the computation of each task. At its extreme, when work in the simulation is divided evenly between nodes with almost no inter-process communication, this is referred to as the embarrassingly parallel approach.

A fine-grained technique will split up work within the inner loops of the simulation. This is necessary when the outer loops proceed in a serial manner. The amount of message passing is much higher in the fine-grained approach, and the benefits of parallelization on distributed memory computers may be less dramatic.

4.1 Requirements and Approach

The Matlab simulation code must be modified as little as possible to run in parallel. Flexibility for future code modifications in Matlab should be maintained. The simulation must maintain the same mathematical models and compute the same numerical results. Performance will be measured by the gain in speed resulting from running the simulation in parallel, as well as the anticipated gain after further increasing the number of processors. Success will also depend on the increased scalability of the simulation for handling large cases, such as the multiconjugate adaptive optics systems that are proposed for ELTs. Such MCAO systems may have on the order of 20,000 WFS subapertures and 10,000 DM actuators.

The code base is Matlab code translated to C using the Matlab C compiler. The parallel modifications will be implemented in a combination of additional Matlab code and C code interfacing the simulation to several optimized parallel packages. The compiled Matlab code is seamlessly interfaced to C allowing future modifications to the Matlab code base to be quickly integrated into the simulation. Finally, all modifications should be compatible. It will be possible to eventually combine all three approaches in one version of the parallel simulation.

4.1 Hardware and Software Implementations

The initial version of the parallel simulation has been developed and tested on a Beowulf cluster located at Gemini Observatory's Northern Operations Center in Hilo. The Beowulf is composed of four nodes, each equipped with a dual-processor Athlon MP. Each machine is configured with a gigabit Ethernet connection and 4 gigabytes of RAM. The cluster runs Red Hat 8.0 using a shared memory Linux kernel and MPICH 1.2.5[12] for message passing. The Petsc[13] and FFTW[14] mathematical packages are used for fine-grained parallel operations.

1. PARALLELIZING THE WAVEFRONT PROPAGATOR

5.1 Coarse-Grained Methods

In many, if not most, adaptive optics concepts proposed for extremely large telescopes, multiple guide stars are used for wavefront sensing. Optical performance must be optimized and evaluated for science objects at different locations across an extended field-of-view. This corresponds in the simulation to simultaneously propagating multiple wavefronts through the atmosphere during any given time quantum. In both geometric and wave optics propagation, the impact of the atmosphere on the individual wavefronts is completely independent. The wavefronts to be propagated at a given time step are divided evenly between the available processors, and the calculations are made individually. At the end of the propagations, the point spread functions for each science object and wavefront sensor measurements from each of the guide stars are collected from the individual nodes. After the wavefront sensor measurements have been

used to generate deformable mirror commands, the mirrors are updated across all nodes for the next set of propagations. The limitation of this method is that the coarse-grained simulation alone cannot scale above the amount of propagations that are evaluated in a specific time step. For common MCAO models, this limits the available parallelization to about 20-25 processors.

The C, Matlab and interface code has been developed to coarsely parallelize the propagations (Figure 2). The entire simulation has been executed and tested in parallel.

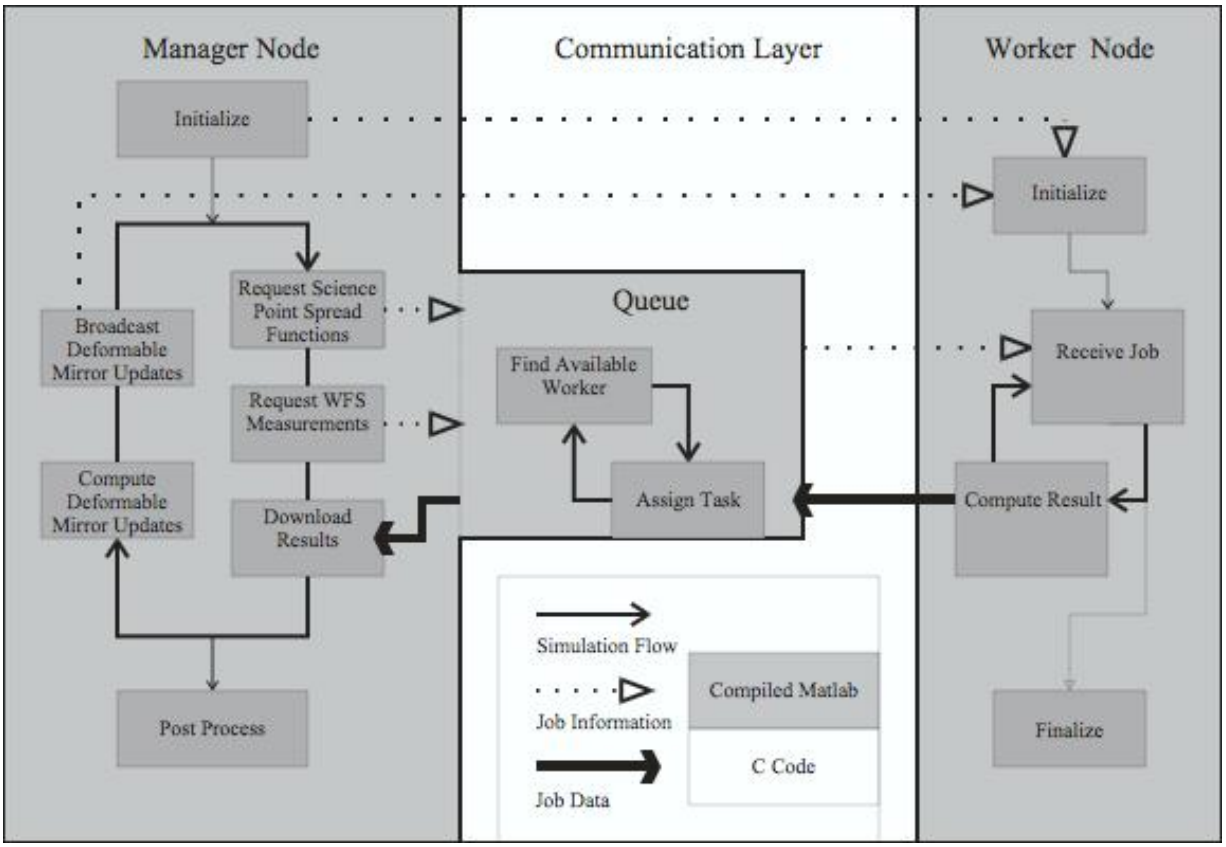
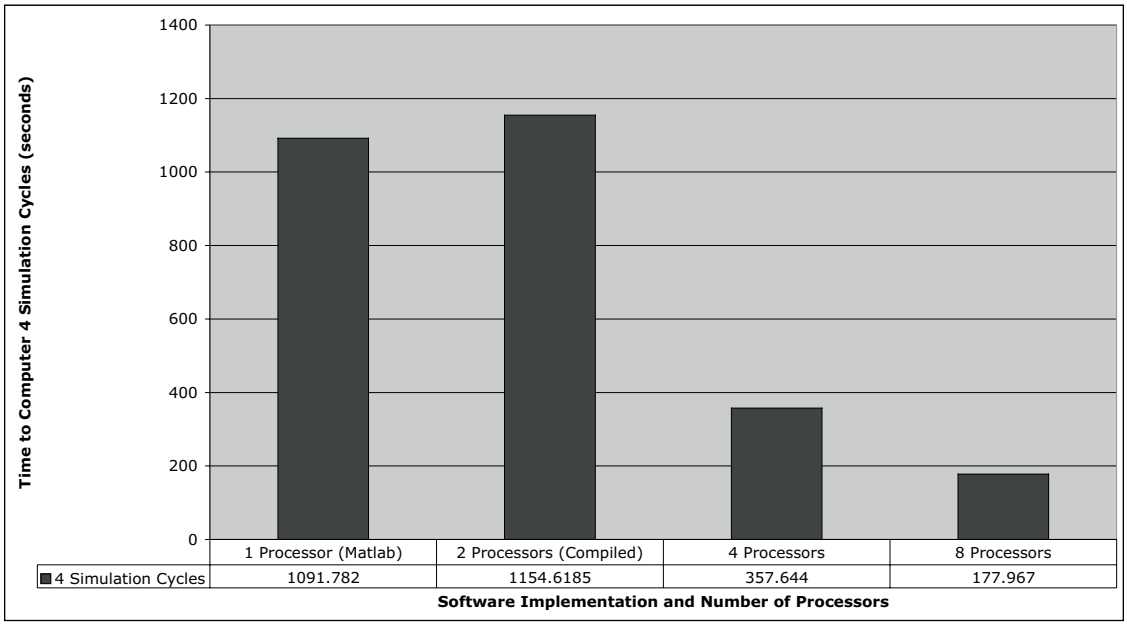


Figure 2: Software Architecture of Original Serial Simulation with Coarse-Grain Parallel Modifications

The compiled coarse-grained simulation was tested for speed against the original Matlab code (Case 1). A simulation for an 8-meter MCAO system was chosen as the sample model. For each surveyed hardware and software implementation, a five-iteration run and a one-iteration run were independently timed. The one-iteration run time was subtracted to provide an estimation of second-per-cycle performance for normal simulations with several hundred or more cycles. The additional overhead of the parallel layout makes the code in this case actually slower than the original simulation when run on a single dual-processor machine. Expanded to four machines with one task on each processor, the parallel simulation demonstrates a gain of 3.1 over the original Matlab simulation. When all four machines are allowed to use their dual-processors, the performance of the cluster running in parallel over the original serial code is 6.1.



Case 1: Fully Implemented Coarse-grained Compiled Simulation versus Serial Matlab Code

5.1 Fast Fourier Transforms

The single most computationally intensive element of adaptive optics simulations is the Fast Fourier Transform (FFT) used to propagate the mathematical representation of each wavefront from phase screen to phase screen. For the standard near-field Fresnel propagator, this operator takes the form:

$$U_{n+1}(\vec{x}) = \Im\left[e^{i\pi\kappa^2\lambda z}\Im^{-1}\left(U_n(\vec{y})S_n(\vec{y})\right)\right](\vec{x})$$

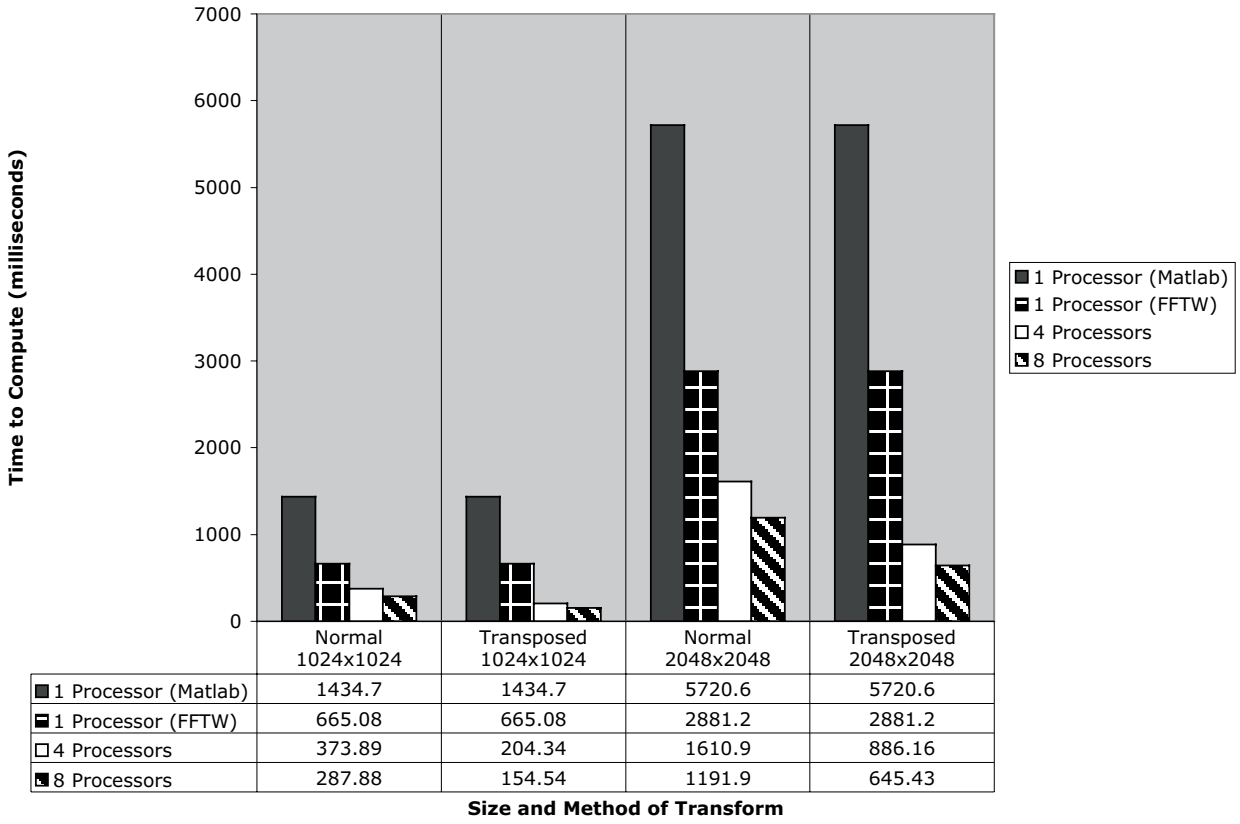
The wavefront next layer (U_{n+1}) is computed by first performing an inverse Fast Fourier Transform, applying the propagator kernel (dependent on wavelength λ, the spatial frequency operator κ, and the distance between layers z), then performing a forward FFT. Additional FFTs may also be used to translate turbulence phase screens with time according to the Taylor–or frozen flow–hypothesis, leading to a requirement of up to three FFT's each time a wavefront is propagated from one screen to the next. The computational burden is increased by the fact that these screens must be significantly larger than the telescope aperture.

Using a fine-grained parallel method to calculate each FFT will reduce the significance of this bottleneck, especially if combined with the coarse-grained method outlined in Section 5.1. Since the FFT is a very common bottleneck in many applications, several optimized software packages are already available. Among these is the whimsically named Fastest Fourier Transform in the West or FFTW[13]. Parallel FFTW is available in C source for both shared and distributed memory versions. The FFTW package is installed on Gemini's Beowulf cluster, and simulations of Fast Fourier Transforms from the inner loops of the propagation have been run to investigate the potential speed gain in the overall simulation.

There is a central difficulty with implementing any version of the two-dimensional FFT in parallel. The computer's representations of each wavefront in memory can be fairly large (64 megabytes for a 2048^2 double-precision, complex-valued wavefront). The time cost of distributing the wavefront to all the processors and collecting it significantly reduces the benefit of performing the transform in parallel. Performing calculations on high-speed networks can reduce some of this latency. The FFTW package also provides some relief by allowing the user to skip a final communication sequence to reorder the solution and instead returns a transposed solution matrix. Inspection of the algorithm used to propagate optical layers reveals that this technique is easily integrated. A pair of Fast Fourier

Transforms, as performed in the propagator, will negate the transposes. The application of the kernel between transforms is a point-wise operation, and therefore unaffected.

The timing results for the Fast Fourier Transforms in Matlab versus Fastest Fourier Transform in the West for 1024x1024 and 2048x2048 sized matrices and using several different options for FFTW are demonstrated in Case 2. It is worth noting that the simulation performance is doubled when FFTW is allowed to leave the solution transposed. The average speed gain for a 2048x2048 transposed Fast Fourier Transform across 8 processors against an identical Matlab transform is 8.9. Assuming that the Fast Fourier Transform dominates 75% of the overall simulation time with no additional communications overhead, this addition to the code alone results in an overall gain of about 3 over current coarse-grained parallel performance.



	Normal 1024x1024	Transposed 1024x1024	Normal 2048x2048	Transposed 2048x2048
1 Processor (Matlab)	1434.7	1434.7	5720.6	5720.6
1 Processor (FFTW)	665.08	665.08	2881.2	2881.2
4 Processors	373.89	204.34	1610.9	886.16
8 Processors	287.88	154.54	1191.9	645.43

Case 2: Parallel FFT Transforms Versus Serial Matlab Code for Two Common Sizes of 2-Dimensional Grids

2. PARALLELIZING THE RECONSTRUCTOR

The function of the wavefront reconstructor in both the simulation and a real-time adaptive optics system is to determine a set of DM actuator commands that will correct the WFS measurements of the turbulence-induced phase distortions. This is mathematically accomplished by finding a solution for a sequential pair of linear inverse problems Ax = By and A'c = B'x – where y is the known wavefront sensor measurement, x is the unknown atmospheric turbulence profile, c is the DM actuator command vector, and all of the matrices A, B, A', and B' are sparse (with low rank perturbations for the case of laser guide star adaptive optics)[10,11]. Existing adaptive optics systems solve for c in terms of y by explicitly computing

$$c = (A'^{-1} B' A^{-1} B)x$$

This direct approach scales very poorly to ELTs due to the time required for explicit matrix multiplications and inversions. More computationally efficient methods are presently under development using a variety of advanced numerical linear algebra techniques[10,11]. In particular, an iterative conjugate-gradient algorithm with multigrid preconditioning and a block-symmetric Gauss-Seidel smoother has been tested independently, and is now ready for implementation in the simulation. Currently, the reconstruction bottleneck lies in the Gauss-Seidel smoothing, where most of the computations take place[10,11].

6.1 Parallelizing the Smoother in Petsc

The Gauss-Seidel smoother finds an approximate solution for the system Ax = r, where A = (A_{ij}) is a symmetric matrix composed of sparse blocks denoted A_{ij}. The approximate solution is derived from the decomposition

$$A = L + D + U$$

where L, D, and U represent the lower triangular, diagonal, and upper triangular blocks of A. Conceptually, the approximate solution is obtained by setting x(0)=0 and performing a small number of iterations of the equations

$$(L + D)x'(n+1) = r - Ux(n)$$
$$(U + D)x(n+1) = r - Lx'(n+1)$$

that are mathematically equivalent to the formulae

$$D_{ii}x_i'(n+1) = r[i] - \sum_{j=i+1}^{N} A_{ij}x_j(n) - \sum_{j=1}^{i-1} A_{ij}x'_j(n+1)$$
$$D_{ii}x_i(n+1) = r[i] - \sum_{j=1}^{i-1} A_{ij}x'_j(n+1) - \sum_{j=i+1}^{N} A_{ij}x_j(n+1)$$

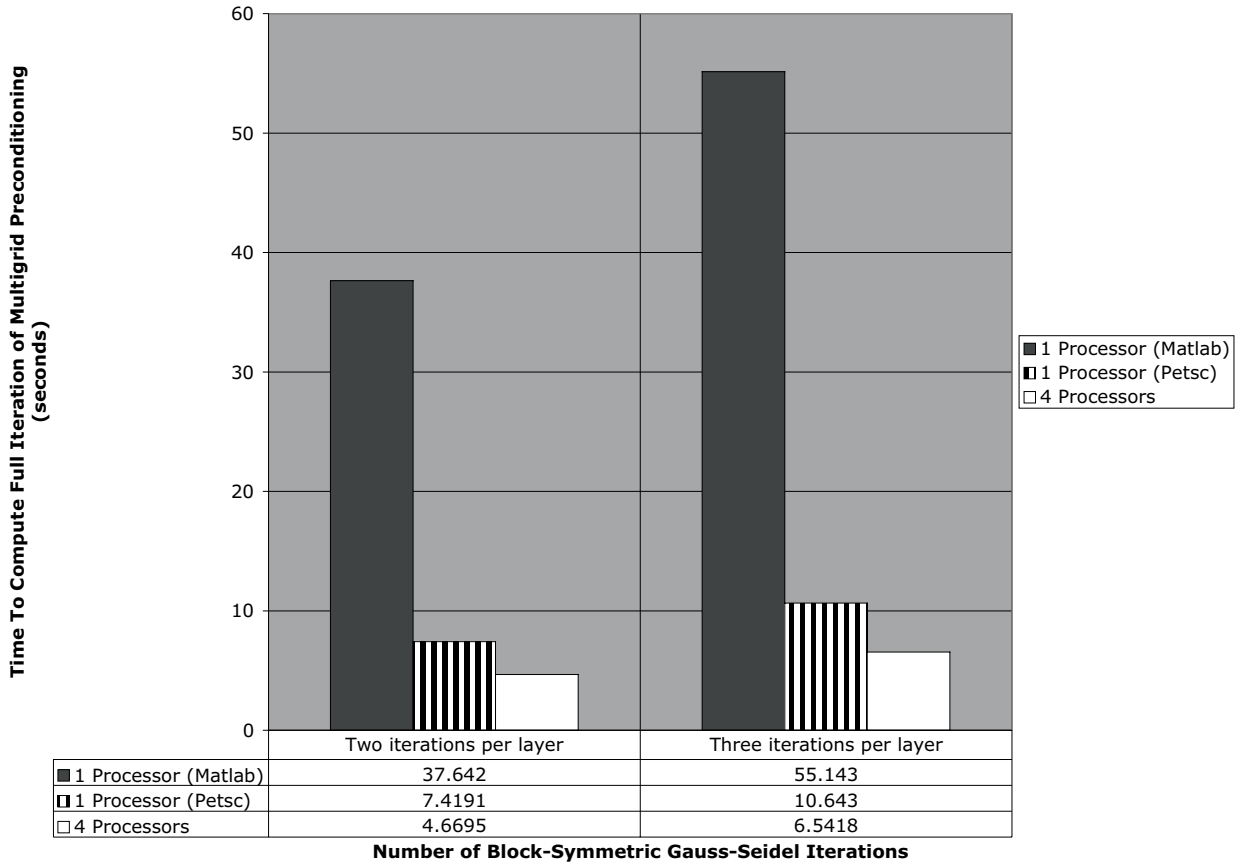
The diagonal blocks D_{ii} are highly sparse, and direct solves are performed using Cholesky factorization and back-substitution. The right-hand sides of these last two equations are computed using sparse matrix-vector multiplies (summed with low-order dense matrix-vector multiplies for the case of laser guide star adaptive optics).

Instead of re-writing all this fine-grain code in C and parallelizing the low-level matrix operations ourselves, an open-source mathematical package, the Portable Efficient Toolkit for Scientific Computation[14] was selected that can perform all of the aforementioned operations in parallel.

All large data structures needed by Petsc are loaded in advance. Only the y vector is needed between iterations of the smoother. C code was written to load the vectors directly from Matlab and iterate through the block-symmetric Gauss-Seidel operations for the smoother. Timing results were generated and checked for numerical accuracy.

6.2 Speed Gain

The timing results for the wavefront reconstructor smoother running in Matlab and in C code interfaced to the parallel Petsc solver are shown (Case 3). These results are for the case of an MCAO system on a 24-meter class ELT with 18,848 WFS measurements, 4,957 DM actuators, and 42,334 phase points. The gain from using Petsc reconstructor in serial over the reconstructor in Matlab is 5.1. The multiplicative gain of 1.6 running Petsc in parallel on four processors against one processor is also shown. The reconstructor running on the 4-node Beowulf cluster executes a resulting 8.3 times faster than the original simulation in Matlab. Additional gain may be implemented by refining the software to take advantage of the dual processors on each node.



Case 3: Parallel Multigrid Iterations Versus Serial Matlab Code

3. CONCLUSION

All the three parallel methods successfully improve the speed of the simulation code on a small 8-node Beowulf Cluster. Of the approaches, only the coarse-grained parallelization has been fully implemented allowing a full simulation to be executed using it. Assuming reasonable scaling from eight processors to the maximum amount for a large-sized multiconjugate adaptive optics model, the parallel simulation will run as much as 15 times faster than its serial counterpart.

For users wishing to take full advantage of the simulation's capabilities on a supercomputer, especially for extremely large telescopes, the additional fine-grained parallel techniques applied to the simulation's Fast Fourier Transforms and wavefront reconstruction linear algebra will enhance the software's speed even further. Extrapolating from the results obtained in Sections 5.1 and 5.2, a multilayered coarse- and fine-grained parallel approach would most likely be capable of approaching gain as great as 50 to 60 on a large Beowulf supercomputer with several hundred nodes. To achieve this speed, the fine-grained approaches need to be completely integrated into the parallel simulation.

The presented parallel software is currently the fastest known full-featured MCAO code. This package will be of interest to the scientists and engineers who are developing the extremely large telescopes of tomorrow and the corresponding adaptive optics systems that provide high-resolution imaging.

ACKNOWLEDGEMENTS

The New Initiatives Office is a partnership between two divisions of the Association of Universities for Research in Astronomy (AURA), Inc.: The National Optical Astronomy Observatory (NOAO) and the Gemini Observatory. NOAO is operated by AURA under a cooperative agreement with the National Science Foundation (NSF). The Gemini Observatory is operated by AURA under a cooperative agreement with the NSF on behalf of the Gemini partnership: the National Science Foundation (United States), the Particle Physics and Astronomy Research Council (United Kingdom), the National Research Council (Canada), CONICYT (Chile), the Australian Research Council (Australia), CNPq (Brazil), and CONICET (Argentina).

REFERENCES

1. S.E. Strom *et al.,* "Giant segmented mirror telescope: A point design based on science drivers.,, In *Future Giant Telescopes*, SPIE Proceedings **4840**, 116-128 (2002).
2. J.E. Nelson, "Progress on the California Extremely Large Telescope (CELT).,, In *Future Giant Telescopes*, SPIE Proceedings **4840**, 47-59 (2002).
3. P. Dierickx *et al*., "Eye of the beholder: Designing the OWL.,, In *Future Giant Telescopes*, SPIE Proceedings **4840**, 151-170 (2002).
4. B.L. Ellerbroek, "Wave optics propagation code for multiconjugate adaptive optics.,, In *Adaptive Optics Systems and Technology II*, SPIE Proceedings **4494**, 104-121 (2001).
5. J.M. Beckers, "Detailed Compensation of Atmospheric Seeing using Multi-Conjugate Adaptive Optics,,, SPIE Proceedings **1114**, 215-217 (1989).
6. D.C. Johnston and B.M. Welsh, "Analysis of multiconjugate adaptive optics,,, J. Opt. Soc. Am. *A* **11**, 394-408 (1994).
7. B.L. Ellerbroek, "First-Order performance evaluation of adaptive-optics systems for atmospheric-turbulence compensation in extended field-of-view astronomical telescopes,,, J. Opt. Soc. Am. A **11**, 783-805 (1994).
8. T. Fusco, J-M. Conan, V. Michau, L.M. Mugnier, and G. Rousset, "Phase estimation for large field of view; application to multiconjugate adaptive optics,,, SPIE Proceedings **3763**, 125-133 (1999).
9. Gemini-South MCAO PDR documentation, <http://www.gemini.edu/sciops/instruments/adaptiveOptics/Aoarchive.html>
10. L. Gilles, B.L. Ellerbroek, and C.R. Vogel, "Layer-oriented multigrid wavefront reconstruction algorithms for multiconjugate adaptive optics,,, SPIE Proceedings **4839**, 1011-1022 (2003).
11. B.L. Ellerbroek, L. Gilles, and C.R. Vogel, "Computationally efficient wavefront reconstruction algorithms for simulations of multiconjugate adaptive optics on giant telescopes,,, SPIE Proceedings **4839** (2002).
12. W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard,,, Parallel Computing **22**, 789-828 (1996).
13. M. Frigo, S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT,,, ICASSP Conference Proceedings **3**, 1381-1384 (1998).
14. S. Balay, W.D. Gropp, L.C. McInnes, and B.F. Smith. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries,,, *Modern Software Tools in Scientific Computing*, E. Arge, A.M. Bruaset, and H.P. Langtangen (eds.), 163-202, Birkhaüser (1997).